- **Code Reflection:** A brief explanation of the code and its purpose, and a brief discussion of your experience in developing it, including any issues that you encountered while completing the exercise and what approaches you took to solve them.
    - This project was meant to organize a csv file into a hash table. It also has the capability to search through to find a specific bid, or delete one from the hash table. Most of the project was fine, took some time to read what I had to work with, then wrote out most of the code. The part I struggled with the most was the remove function, it was longer than I expected it to be and I kept attempting to make it shorter than I suppose it needed. Also, the UINT_MAX creating a special case for the head node was needlessly complex and incredibly confusing.

- **Pseudocode or Flowchart:** A pseudocode or flowchart description of the code that is clear and understandable and captures accurate logic to translate to the programming language.

```
START main ()

INIT Hash Table

PRINT menu

IF case "1":

     START clock()

     CALL loadBids:

     END clock ()

     PRINT time in seconds

     BREAK

IF case "2":

     CALL PrintAll():

          FOR EACH bucket :

               IF the "key" doesn't equal the "head":

                    PRINT

                    node = the next bucket

                    WHILE the node is not empty:

                         PRINT

                    node = the next node

     BREAK

IF case "3":

     PRINT "Insert desired bid ID: "
```

```
     GET “bidId”

     START clock()

     bid = CALL Search(“bidId”):

          INIT unsigned key as hash()

          node  = buckets at “key”

          IF node “key” IS NOT EQUAL to the “head” node AND node bidId is
same as the search bidId:

               RETURN THIS bid

          IF node “key” IS EQUAL to the “head” node (UINT_MAX):

               RETURN bid (bid will only have the bidId but nothing else)

          WHILE node “key” IS NOT NULL:

               IF node bidId is the same as search bidId:

                    RETURN node bid

               node = next node

          RETURN bid

     END clock

     IF bid, CALL isValid():

          CALL displayBid(bid):

               PRINT

     ELSE:

          PRINT “Bid not found”

     PRINT time in seconds

     BREAK

IF case “4”:

     PRINT “Insert bid ID”

     GET “bidId”

     CALL Remove(“bidId”):

          INIT unsigned int key AS hash()
```

```
          node = buckets at “key”
          IF node “key” IS NOT EQUAL to “head” node:
               IF node bidId is the same as “bidID”:
                    IF next node is NULL:
                         Node key = head node
                    ELSE:
                         Buckets at key = pointer node next
               ELSE:
                    INIT current node AS next node
                    INIT previous node AS node
                    WHILE current IS NOT NULL:
                         IF current bidID is the same as “bidId”:
                              Next node after previous = next node after
current
                              DELETE current
                              RETURN
                         Previous = current
                         Current = next node after current
     BREAK
PRINT “Good bye.”
```